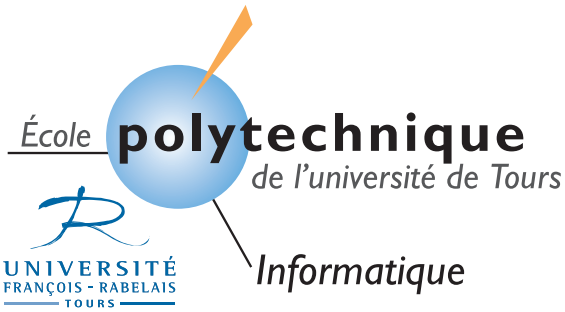

École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
Fax +33 (0)2 47 36 14 22
www.polytech.univ-tours.fr

Spécialité Informatique Industrielle
4ème année
2014-2015

Reconnaissance de grille

Projet de développement embarqué

Apprenti :
Alexandre BILLAY, Thibault ARTUS
alexandre.billay@etu.univ-tours.fr, theskull-machine@gmail.com

Tuteur :
Yannick KERGOSIEN
yannick.kergosien@univ-tours.fr
Polytech'Tours

Version du 15 janvier 2015

Table des matières

Table des figures

Introduction

Dans le cadre de notre quatrième année au sein de Polytech'Tours, nous avons dû réaliser un projet de développement embarqué sur une durée égale à 4 mois. Après 2 heures de présentation des sujets, nous avons choisi le développement et intégration d'un système de reconnaissance de grille sur tablette Androïd. Ce projet découle d'un PFE (Projet de Fin d'Étude), le Stacker Crane Probleme interprété par un pont roulant, réalisé lors l'année précédente par Thibault Morelle. Quant à notre projet, il fut décidé de reprendre la partie du PFE sur la reconnaisance automatique des objets à déplacer par simple prise de photo. Cette partie ne fonctionnait pas. Notre projet a dû être fait en collaboration avec Clément Laloubeyre, un élève en cinquième année qui a pris la suite du PFE de Thibault Morelle.

CHAPITRE 1

Conduite du projet

1.1 Cahier des charges

L'objectif du projet est de concevoir un module de détection d'une grille et de localisation de deux types de pièces de couleurs dans cette grille à l'aide de la caméra d'une tablette Android. Ce module sera à intégrer dans une application mobile permettant de contrôler un pont roulant ayant pour but de déplacer les objets détectés dans cette grille.

1.2 État des lieux

Lors de notre reprise du projet, l'application permettait la détection des formes et des couleurs d'une image créée sous *Paint* alors qu'une image prise à l'aide de la caméra de la tablette ne permettait pas ce fonctionnement.

1.3 Chronogramme réel du projet

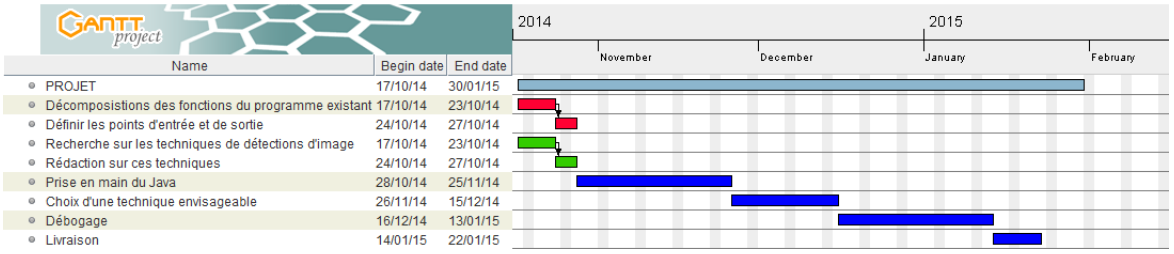

FIGURE 1.1 – Diagramme de Gantt

Après notre premier entretien avec Clément, nous avons réalisé tout au long du projet le diagramme de Gantt réel.

Au début du projet, nous avons fait nos tâches en parallèle. La décomposition des fonctions du programme existant a été réalisé par Thibault ainsi que leurs définition des points d'entrée et sortie (en rouge). La recherche sur les techniques de détections d'image et la rédaction de synthèse sur ces techniques a été produite par Alexandre (en vert). Le reste du projet, a été fait ensemble, c'est à dire la prise en main du Java ainsi que les logiciels attachés, l'analyse d'une technique implémentable envisageable, le débogage et la livraison de l'application.

1.4 Software utilisés

Lors de ce projet nous avons pu mettre en pratique différents software :

- **ADT Bundle** (IDE Eclipse 4.2 + SDK Manager) : compatible Java 7. La version de l'application Androïd est la 4.3.

FIGURE 1.2 – Android Developer Tools

- **Plugin Lejos** pour Eclipse : nécessitant l'installation de drivers pour les briques NXT.

FIGURE 1.3 – Plugin Lejos pour Eclipse

- **Genymotion** : émulateur plus performant que celui inclus à l'ADT. Nous avons pu simuler la quasi totalité de nos actions sur une Galaxy Tab 3 émulée.

FIGURE 1.4 – Émulateur Androïd Genymotion

CHAPITRE 2

Analyse des classes

Dans ce chapitre, nous présentons les classes Hough.java, PictureHandler.java et HoughView sur lesquelles nous avons dû travailler.

2.1 Hough.java

2.1.1 Constructeur

public Hough(int width, int height)

Points d'entrée :

- width : largeur de l'image ; entier
- height : hauteur de l'image ; entier

2.1.2 Méthodes (Transformée de Hough)

public void vote(int x, int y)

Points d'entrée :

- x : largeur de l'image/2 ; entier
- y : hauteur de l'image/2 ; entier

public List<double[]> getWinners(int threshold, int radius) ; on récupère la valeur extreme de la transformée de Hough

Points d'entrée :

- threshold : seuil de l'image
- radius : rayon

Point de sortie :

- winners : tableau contenant les valeurs extremes de Rho et Théta ; tableau de réels

private int distance(int r0, int t0, int r1, int t1)

Points d'entrée :

- r0 : point 0 de Rho ; entier
- t0 : point 0 de Théta ; entier
- r1 : point 1 de Rho ; entier
- t1 : point 1 de Théta ; entier

Point de sortie :

- dist : Retourne la valeur minimale entre dist et le maximum de la valeur absolue entre (r0-r1) et (t0-t1) ; entier

2.1.3 Méthodes (Conversions)

public int RhoToIndex(double rho)

Points d'entrée :

- rho : réel

Point de sortie :

- On retourne un entier qui est spécialement converti pour rentrer dans notre index (matrice de valeur de Rho)

public double IndexToRho(int index)

Points d'entrée :

- index : entier

Point de sortie :

- On retourne un reel qui vient de la conversion d'un entier (Rho) de la matrice d'index.

public int ThetaToIndex(double theta)

Point d'entrée :

- theta : réel

Point de sortie :

- On retourne un entier qui est spécialement converti pour rentrer dans notre index (matrice de valeur de Theta)

public double IndexToTheta(int index)

Point d'entrée :

- index : entier

Point de sortie :

- On retourne un réel qui vient de la conversion d'un entier (Theta) de la matrice d'index.

public double[] rhotheta_to_ab(double rho,double theta) : conversion de rho et theta pour permettre son utilisation dans une équation de droite Y=a*X+b

Point d'entrée :

- rho : reel
- theta : reel

Point de sortie :

- a, b : reel

2.1.4 Accesseurs

public int getMaxIndexTheta()

Point de sortie :

– maxIndexTheta : entier ; on récupère la valeur maximale de theta dans l'index

public int getMaxIndexRho()

Point de sortie :

– maxIndexRho : entier ; on récupère la valeur maximale de rho dans l'index

public int[][] getAccumulator()

Point de sortie :

– acc : tableau 2 dimensions d'entier ; on récupère ? ? ?

2.2 PictureHandler.java

2.2.1 Constructeur

public PictureHandler(PhotoFragment cxt, int callerId)

Points d'entrée :

– cxt : photo du parent ; PhotoFragment
– callerId : entier

2.2.2 Méthodes

public void onPictureTaken(byte[] data, Camera camera) ; decode de l'image Bitmap

Points d'entrée :

– data : tableau d'octets
– camera : Camera

protected void onPreExecute() : pour chaque nouvelle ligne, on créé un nouvel HashMap les contenant

protected Void doInBackground(Bitmap... pictureFile) : dans cette méthode, on charge l'image enregistré puis on éxécute la transformée de Hough puis l'extraction des lignes de l'image.

Points d'entrée :

– pictureFile : type Bitmap

protected void onPostExecute(Void result)

private void doTH(Bitmap img0) : application de l'algorithme de Hough sur l'image

Points d'entrée :

– img0 : type Bitmap

private void doLinesExtraction(Bitmap img0) : permet de faire l'extraction des lignes suite à la transformée de Hough

Points d'entrée :

– img0 : type Bitmap

private void sendLinesToDrawToUiThread(HashMap<Integer, ArrayList<Point» lines) : Permet de dessiner les lignes stockées dans le Hashmap.

Points d'entrée :

– lines : type HashMap<Integer, ArrayList<Point» ; Un tableau qui a comme clef des entiers permettant de retrouver plus facilement les listes de points de chaque ligne précédement stockée.

2.3 HoughView.java

CHAPITRE 3

Causes des dysfonctionnements possibles

Au fur et à mesure des analyses faites sur l'application, en croisant les résultats des images réelles avec celles créées sous Paint, nous en sommes arrivés à 4 causes possibles de dysfonctionnement :

1. Les photos prises par la tablette ont des couleurs mal prises en charge par l'algorithme de détection de couleur, les filtres fonctionneraient mal pour isoler les lignes noires.
2. La transformée de Hough serait faussée et/ou pas assez précise. Nous avons fait des recherches sur des alternatives possibles décrites dans le chapitre suivant.
3. La mauvaise gestion du format. Les images scrutées par l'algorithme sont de format différent (Bitmap, jpeg et png).
4. Les photos prises avec la caméra sont trop inclinées.

CHAPITRE 4

Recherches

4.1 Détection de formes

4.1.1 Tranformée de Hough

Le principe qui sous-tend la transformée de Hough est qu'il existe un nombre infini de lignes qui passent par un point, dont la seule différence est l'orientation (l'angle). Le but de la transformée est de déterminer lesquelles de ces lignes passent au plus près du schéma attendu.

Dans la transformée de Hough, dite aussi transformée standard de Hough ou SHT, chaque ligne est un vecteur de coordonnées paramétriques :

- θ : l'angle
- ρ : la norme du vecteur (la longueur du segment perpendiculaire à la droite d'angle θ et passant par l'origine)

En transformant toutes les lignes possibles qui passent par un point, c'est-à-dire en calculant la valeur de ρ pour chaque θ, on obtient une sinusoïde unique appelée espace de Hough. Si les courbes associées à deux points se coupent, l'endroit où elles se coupent dans l'espace de Hough correspond aux paramètres d'une droite qui relie ces deux points.

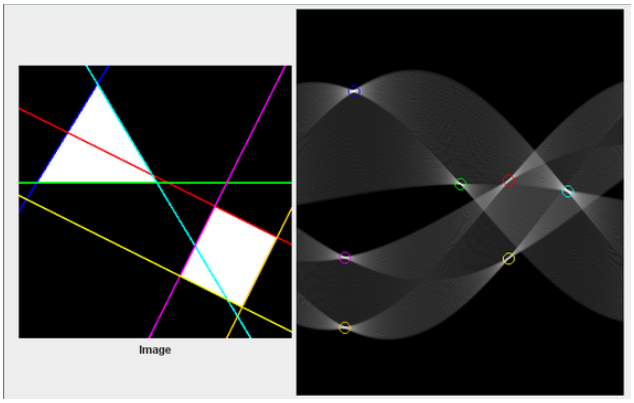

FIGURE 4.1 – Transformée de Hough

4.1.2 Codage de Freeman absolu

Codage avec un nombre limité de bits de la direction locale d'un élément de contour défini dans une image discrète, puis constitution d'une chaine de codes à partir d'un pixel initial, considérant qu'un élément de contour relie 2 pixels connexes.

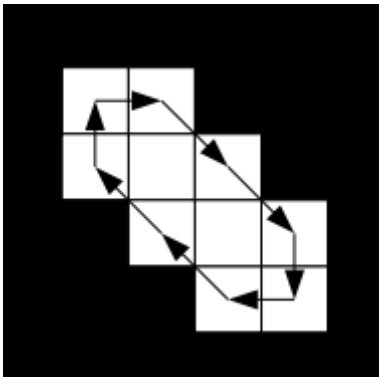

FIGURE 4.2 – Constitution d'une chaîne de codes

4.1.3 Codage de Freeman relatif

Dans cette variante on code le changement de direction plutôt que de la direction.

FIGURE 4.3 – Codage du changement de direction

Le code de Freeman standard est invariant en translation uniquement. Le code Freeman relatif est invariant en translation et aux rotations de 45°.

Codage sur 2 bits pour connexité 4. Codage sur 3 bits pour connexité 8. Codage sur 4 bits pour connexité 8 + longueur 2. Etc...

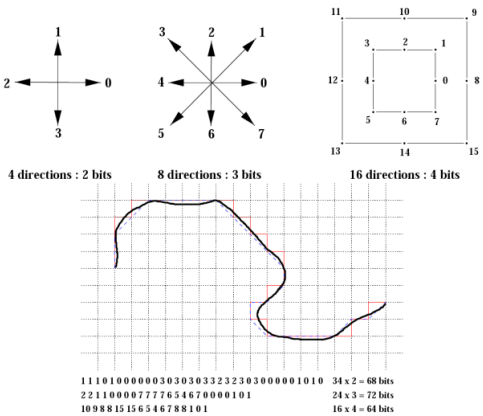

FIGURE 4.4 – Nombre de directions

4.1.4 Régression linéaire

On approche un ensemble de points par un segment de droite. Pour cela on minimise un résidu entre le modèle (la droite) et les données (points repérés par leurs coordonnées).

– Résidu :

$$d^2(a, b) =$$

4.2 Détection de couleur

CHAPITRE 5

Travaux effectués

Relecture et compréhension du code
Apprentissage des bases du Java
Mise en place de l'IDE et des librairies
Identification des entrées/sorties de la transformée
Recherche d'algorithmes alternatifs
Réécriture des filtres en niveau de gris
Tests en mode débug avec Genymotion
Réécriture d'une transformée de Hough en standalone

CHAPITRE 6

Démonstration

Conclusion

Projet de développement embarqué

Spécialité Informatique Industrielle
4ème année
2014-2015

Reconnaissance de grille

Résumé:

Mots clefs:

Apprenti :
Alexandre BILLAY, Thibault ARTUS
alexandre.billay@etu.univ-tours.fr, theskull-machine@gmail.com

Tuteur :
Yannick KERGOSIEN
yannick.kergosien@univ-tours.fr
Polytech'Tours